Managing Linguistic Data Summaries in Advanced P2P Applications

Rabab Hayek, Guillaume Raschia, Patrick Valduriez, Noureddine Mouaddib

Managing Linguistic Data Summaries in Advanced P2P Applications

Rabab Hayek, Guillaume Raschia, Patrick Valduriez and Noureddine Mouaddib

Abstract As the amount of stored data increases, data localization techniques become no longer sufficient in P2P systems. A practical approach is to rely on compact database summaries rather than raw database records, whose access is costly in large P2P systems. In this chapter, we describe a solution for managing linguistic data summaries in advanced P2P applications which are dealing with semantically rich data. The produced summaries are synthetic, multidimensional views over relational tables. The novelty of this proposal relies on the double summary exploitation in distributed P2P systems. First, as semantic indexes, they support locating relevant nodes based on their data descriptions. Second, due to their intelligibility, these summaries can be directly queried and thus approximately answer a query without the need for exploring original data. The proposed solution consists first in defining a summary model for hierarchical P2P systems. Second, appropriate algorithms for summary creation and maintenance are presented. A query processing mechanism, which relies on summary querying, is then proposed to demonstrate the benefits that might be obtained from summary exploitation.

Rabab Hayek
LINA, 2 rue de la Houssiniere 44322 Nantes France, e-mail: rabab.hayek@univ-nantes.fr

Guillaume Raschia
LINA, 2 rue de la Houssiniere 44322 Nantes France, e-mail: guillaume.raschia@univ-nantes.fr

Patrick Valduriez
INRIA, 2 rue de la Houssiniere 44322 Nantes France, e-mail: Patrick.Valduriez@inria.fr

Noureddine Mouaddib
LINA, 2 rue de la Houssiniere 44322 Nantes France, e-mail: Noureddine.Mouaddib@univ-nantes.fr

1 Introduction

As revealed by the large number of P2P research papers and surveys (including the previous chapter), data localization has been the main issue addressed in the P2P community. However, advanced data management techniques, which allow more than locating data, are strongly required to support database applications. These applications are facing the *information explosion* problem: a huge amount of information is generated and stored each day into data sources (e.g. biological, astronomical database applications). This ever increasing amount of available data, in addition to the high distribution of connected resources, makes search techniques no more sufficient for supporting P2P database applications. To illustrate, in a scientific collaborative application, a doctor may require information about patients diagnosed with some disease, without being interested in individual patient records. Besides, in today's decision-support applications, users may prefer an approximate but fast answer, instead of waiting a long time for an exact one.

To address the problem of managing voluminous databases, the data summarization paradigm has emerged into the database field. The objective of data summarization is to synthesize the information which is disseminated within large datasets, in order to provide the *essential*. Obviously, data summarization is of higher importance in P2P systems. In these systems, the participants share a global database which is equivalent to the aggregation of all their local databases, and thus may become much more voluminous. Therefore, *Reasoning on compact data descriptions that can return approximate answers like "dead Malaria patients are typically children and old" to queries like "age of dead Malaria patients", is much more efficient than retrieving raw records, which may be very costly to access in highly distributed P2P databases.*

The authors in [21] propose a new solution for managing linguistic data summaries in P2P systems. The produced summaries are synthetic, multi-dimensional views over relational tables. The novelty of this proposal relies on the double exploitation of summaries in distributed P2P systems. First, as semantic indexes, they support locating relevant nodes based on their data descriptions. Second, due to their intelligibility, these summaries can be directly queried to approximately answer a query without the need for exploring original data, which might be highly distributed in P2P systems.

This work makes the following contributions. Section 2 first proposes a summarization process that exhibits many salient features making from it an interesting process to be integrated into P2P environments. In Section 3, a summary model is defined in the context of hierarchical P2P systems. The network is organized into *domains*, where a domain is defined as being the set of a supernode and its associated leaf nodes. In a given domain, peers cooperate to maintain a global summary over their shared data. Section 4 discusses the gains that might be obtained in query processing from the use of data summaries. In Section 5, the performance of the summary proposal is

evaluated through a cost model and a simulation model. Simulation results show that the cost of query routing is significantly reduced without incurring high costs of summary updating. Section 6 discusses related works, and Section 7 concludes.

2 Summarization Process

In this section, we first briefly discuss the existing approaches of data summarization according to some required characteristics. Then, we define the input data and describe the summarization process of the approach adopted in [21]. Finally, the structure of distributed summaries in P2P systems is formally defined.

2.1 Data Summarization

The data summarization paradigm has been extensively studied in centralized environments to address the problem of managing voluminous data sets. In the literature, many approaches have been proposed, each satisfying the requirements of specific applications. Here, we discuss the characteristics that should control the choice of a data summarization process in order to satisfy the purposes discussed in our introduction.

- **Compression:** the summarization process should provide compact versions of the underlying data. However, we are not referring here to (syntactic) compression techniques. These techniques consider a data source as a large byte string and thus use compression algorithms such as Huffman or Lempel-Ziv coding. They were mainly proposed to deal with throughput and space storage constraints. In fact, we are concerned with compression (or data reduction) techniques that can provide fast and approximate answers. This is very efficient in the context where obtaining an exact answer is a time-consuming process, and an approximate answer may give information that are sufficiently satisfying. Examples of such techniques are discrete wavelet transform and linear regression used in signal processing [19], sampling and histograms [14] used in statistics. Index trees such as B^+Tree [8] and K-d-Tree [6] could also be considered as data reduction techniques in the database field. These indexes do not provide approximate answers, however, they allow to optimize and accelerate the access within a large data set.

- **Intelligibility:** the summarization process should synthesize the information disseminated within large datasets in order to provide the *essential*. In fact, the data summarization techniques are data reduction techniques

that are supposed to provide intelligible representations of the underlying data. Three categories of database summarization techniques have been proposed in the literature. The first one focuses on aggregate computation. Examples are OLAP and multidimensional databases which allow an end-user to query, visualize and access part of the database using cubes of aggregate values computed from raw data [5]. The second category, so-called semantic compression (SC), deals with intentional characterization of groups of individuals to provide higher-level models such as decision trees or association rules. Examples are [24], [12] which explicitly refer to semantic compression of structured data.
The last category is interested in metadata-based semantic compression (MDBSC) approaches. These approaches use metadata to guide the compression process. The produced summaries are highly comprehensible since their descriptions rely on user-defined vocabularies. The main difference between SC and MDBSC is that the latter provide data descriptions which precisely fit the user perception of the domain, whereas the former aims to identify hidden patterns from data. One representative of the MDBSC approaches is the Attribute-Oriented Induction process (AOI). It provides reduced versions of database relations using *is-a* hierarchies, i.e. a concept tree built over each attribute domain [11].

- **Robustness:** the summarization process should handle the vagueness and the imprecision inherent to natural language. The theory of fuzzy sets provides a formal framework associated with a symbolic/numerical interface using linguistic variables [17] and fuzzy partitions [23]. Hence, the linguistic summaries have the advantages of being robust and formulated in a user-friendly language (i.e. linguistic labels).

- **Scalability:** the summarization process should be scalable in terms of the amount of processed data. It should be able to treat voluminous databases with low time complexity, and controlled memory consumption. Besides, an important issue is the ability of the process to be parallelized and distributed among multiple processors or computers.

In [21], the approach used for data summarization is based on SaintEtiQ [22]: an online linguistic approach for summarizing databases. The SaintEtiQ model aims at apprehending the information from a database in a synthetic manner. This is done through generating linguistic, multidimensional summaries that are arranged and incrementally maintained in a hierarchy. The hierarchies of SaintEtiQ summaries differ from those obtained by the *is-a* approach in that they rely on one set of linguistic terms, without level assignment. Indeed, the SaintEtiQ model proceeds first in an abstraction of data by the use of a user-defined vocabulary, and then in performing a classification that produces data descriptions at different levels of granularity (i.e. levels of abstraction). The SaintEtiQ process exhibits many interesting

features and is proposed to be efficiently integrated into P2P environments, as it is revealed in the rest of this section.

2.2 Input Data

The summarization process takes as input the original data to be summarized, and the *"Background Knowledge"* BK which guides the process by providing information about the user's perception of the domain.

2.2.1 Data Model

The data to be summarized come from relational databases. As such, the data are organized into records, with a schema $R(A_1, A_2, \ldots, A_n)$. Each attribute A_i is defined on an attribute domain D_i, which may be numeric or symbolic. Thus, each tuple t consists of n attribute values from domains D_1 to D_n. It is given by:

$$t = \langle t.A_1, t.A_2, \ldots, t.A_n \rangle$$

A constraint on these data is that it should be complete: any value $t.A_i$ is necessarily known, elementary, precise, and certain. In other terms, all records with a null value are dismissed.

2.2.2 Background Knowledge

A unique feature of the summarization system is its extensive use of a *Background Knowledge (BK)*, which relies on linguistic variables and fuzzy partitions.

Consider the following relational database of a given hospital, which is reduced to a single *Patient* relation (Table 1)[1]. Figure 1 shows a linguistic variable defined on the attribute AGE where descriptor YOUNG ADULT is defined as being plainly satisfactory to describe values between 19 and 37 and less satisfactory as the age is out of this range. Similarly, Figure 2 provides the linguistic variable defined on attribute BMI[2]. Thus, linguistic variables come with linguistic terms (i.e. descriptors) used to characterize domain values and, by extension, database tuples. For a continuous domain D_i, the linguistic variable is a fuzzy partition of the attribute domain. For a discrete domain D_i (DISEASE and SEX in our example), the BK element is a fuzzy set of nominal values. In short, the BK supports the summarization process with means to match attribute domain values with the summary expression vocabulary.

[1] Patient day: number of days spent in the hospital.

[2] BMI is the patient's body weight divided by the square of its height.

Note that the BK, given by users or experts of the data domain, concerns the attributes which are considered as pertinent to the summarization process. In our example, we have excluded the *patient day* attribute. However, when the descriptions of that attribute values might be useful for the hospital application (i.e. requiring information about the *length of stay* of patients), a corresponding fuzzy partition should be also provided.

Table 1 Raw data

Id	Age	Sex	BMI	Patient days	Disease
t_1	16	*female*	16	350	*Anorexia*
t_2	60	*male*	32	4	*High blood pressure*
t_3	18	*female*	17	280	*Anorexia*
t_4	17	*female*	18	230	*Anorexia*
t_5	54	*female*	26	14	*Osteoporosis*

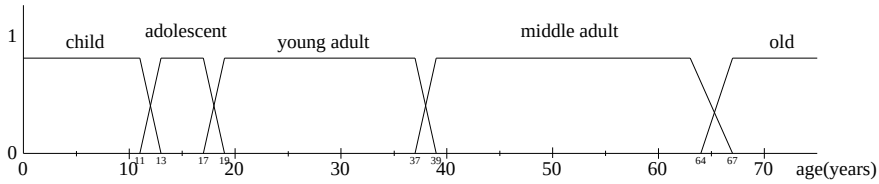

Fig. 1 Fuzzy Linguistic Partition on **age**

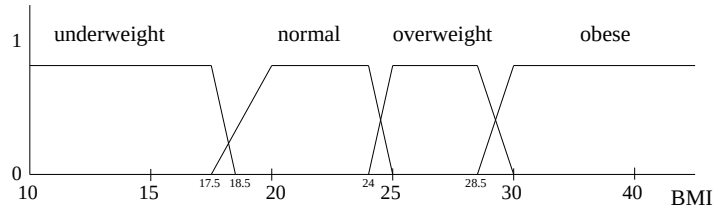

Fig. 2 Fuzzy Linguistic Partition on **BMI**

2.3 Process Architecture

A service oriented architecture has been designed for the summarization process in order to incrementally build data summaries. By "incremental", we

mean that the database tuples are processed one by one, and the final hierarchy of summaries is obtained by repeating this summarization process for each tuple in the table at hand. The architecture is organized into two separate services: online mapping and summarization.

2.3.1 Mapping Service

The mapping service takes as input the original relational records and performs an abstraction of the data using the BK. A representation of the data under the form of fuzzy sets is obtained. Basically, the mapping operation replaces the original attribute values of every record in the table by a set of linguistic descriptors defined in the BK. For instance, with the linguistic variable defined on the attribute AGE (Figure 1), a value $t.AGE = 18$ years is mapped to $\{0.5/adolescent, 0.5/young\ adult\}$ where 0.5 is a membership grade that tells how well the label *young adult* describes the value 18. Extending this mapping to all the attributes of a relation could be seen as locating the overlapping cells in a grid-based multidimensional space which maps records of the original table. The fuzzy grid is provided by the BK and corresponds to the user's perception of the domain.

In our example, tuples of Table 1 are mapped into four distinct grid-cells denoted by c_1, c_2, c_3, c_4, and c_5 in Table 2. The fuzziness in the vocabulary definition of *BK* permits to express any single value with more than one fuzzy descriptor and thus avoid threshold effect thanks to the smooth transition between different categories. For instance, the tuple t_3 of Table 1 is mapped to the two grid cells c_1 and c_2 since the value of the attribute *age* is mapped to the two linguistic terms: *adolescent* and *young adult*. Similarly, the tuple t_4 is mapped to c_1 and c_3 since the value of the attribute *BMI* is mapped to $\{0.7/underweight, 0.3/normal\}$. The tuple count column gives the proportion of records that belongs to the cell, and $0.5/young\ adult$ says that *young adult* fits the data only with a degree of 0.5. The degree of a label is the maximum of membership grades to that label, computed over all the tuples in the corresponding grid cell. For instance, the degree of *adolescent* in c_1 is equal to one, which is the maximum grade value of the two tuples t_1 and t_3.

Table 2 Grid-cells mapping

Id	Age	Sex	BMI	Disease	tuple count
c_1	*adolescent*	*female*	*underweight*	*Anorexia*	2.2
c_2	$0.5/young\ adult$	*female*	*underweight*	*Anorexia*	0.5
c_3	*adolescent*	*female*	$0.3/normal$	*Anorexia*	0.3
c_4	*middle adult*	*male*	*obese*	*High blood pressure*	1
c_5	*middle adult*	*female*	*overweight*	*Osteoporosis*	1

Therefore, the BK leads to the point where tuples become indistinguishable and then are grouped into grid-cells such that there are finally many more records than cells. Every new (coarser) tuple stores a record count and attribute-dependent measures (min, max, mean, standard deviation, etc.). It is then called a *summary*.

2.3.2 Summarization Service

The summarization service is the last and the most sophisticated step of the SaintEtiQ system. It takes *grid-cells* as input and outputs a collection of summaries hierarchically arranged from the most generalized one (the root) to the most specialized ones (the leaves) [22]. Summaries are clusters of grid-cells, defining hyperrectangles in the multidimensional space. In the basic process, leaves are grid-cells themselves and the clustering task is performed on K cells rather than N tuples ($K << N$).

From the mapping step, cells are introduced continuously in the hierarchy with a top-down approach inspired of D.H. Fisher's Cobweb [16], a conceptual clustering algorithm. Then, they are incorporated into best fitting nodes descending the tree. Three more operators could be apply, depending on partition's score, that are *create*, *merge* and *split* nodes. They allow developing the tree and updating its current state. Figure 3 represents the summary hierarchy built from the cells c_1, c_2, c_3, c_4, and c_5.

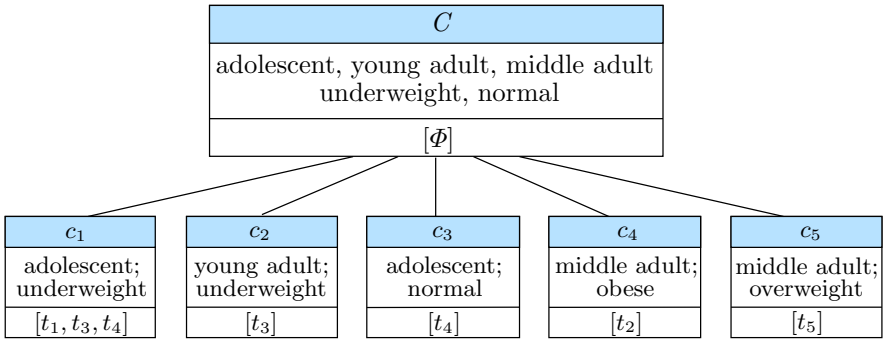

Fig. 3 Example of SaintEtiQ hierarchy

The SaintEtiQ process has been successfully integrated to an existing DataBase Management System (DBMS) thanks to the design of a service-oriented architecture where service calls are made through the SOAP protocol. On the other hand, the *eXtensible Markup Language* XML is the format adopted for exchanging data with the DBMS, as well as for summary representation all along the summarization process (i.e. raw data, BK, grid cells

and the final summaries are represented and exchanged under the form of XML documents).

2.3.3 Scalability Issues

Memory consumption and time complexity are the two main factors that need to be taken care of in order to guaranty the capacity of the summary system to handle massive datasets. First, the time complexity of the SaintEtiQ process is in $O(n)$, where n is the number of candidate tuples to incorporate into a hierarchy of summaries. However, the number of candidate tuples that are produced by the mapping service is dependent only on the fuzziness of the BK definition. A crisp BK will produce exactly as many candidate tuples as there are original tuples. Besides, an important feature is that in the summarization process, raw data have to be parsed only once, and this is performed with a low time cost. Second, the system requires low memory consumption for performing the summary construction algorithm as well as for storing the produced summaries. Moreover, a cache manager is in charge of summary caching in memory and it can be bounded to a given memory requirement. Usually, less than a hundred of summaries are needed in the cache since this number covers the two or three top levels of even a wide hierarchy. Least recently used summaries are discarded when a required summary is not found in the cache.

On the other hand, the parallelization of the summary system is a key feature to ensure smooth scalability. The implementation of the system is based on the Message-Oriented Programming paradigm. Each sub-system is autonomous and collaborates with the others through disconnected asynchronous method invocations. It is among the least demanding approaches in terms of availability and centralization. The autonomy of summary components allows for a distributed computing of the summary process. Once a component completes the treatment and evaluates the best operator for the hierarchy modification, if needed, a similar method is successively called on children nodes. The cache manager is able to handle several lists of summaries residing on different computers [22].

2.4 Distributed Summary Representation

In this section, we introduce basic definitions related to the summarization process.

Definition 1. Summary Let $E = \langle A_1, \ldots, A_n \rangle$ be a n-dimensional space equipped with a grid that defines basic n-dimensional areas called cells in E. Let R be a relation defined on the cartesian product of domains D_{A_i} of

dimensions A_i in E. Summary z of relation R is the bounding box of the cluster of cells populated by records of R.

The above definition is constructive since it proposes to build generalized summaries (hyper-rectangles) from cells that are specialized ones. In fact, it is equivalent to performing an *addition* on cells:

$$z = c_1 + c_2 + \ldots + c_p$$

where $c_i \in L_z$, the set of p cells (summaries) covered by z.

A summary z is then an *intentional description* associated with a set of tuples R_z as its *extent* and a set of cells L_z that are populated by records of R_z.

Thus, summaries are areas of E with hyper-rectangle shapes provided by BK. They are nodes of the summary tree built by the SaintEtiQ system.

Definition 2. Summary Tree A summary tree is a collection S of summaries connected by $\preccurlyeq$, the following partial order:

$$\forall z, z' \in \mathcal{Z}, \quad z \preccurlyeq z' \Longleftrightarrow R_z \subseteq R_{z'}$$

The above link between two summaries provides a generalization/specialization relationship. And assuming that summaries are hyper-rectangles in a multi-dimensional space, the partial ordering defines *nested summaries* from the larger one to the single cells. General trends in the data could be identified in the very first levels of the tree whereas precise information has to be looked at near the leaves.

In [21], a summary tree is also considered as an indexing structure over distributed data in a P2P system. Thus, a new dimension has been added to the definition of a summary node z: a *peer-extent* P_z, which provides the set of peers having data described by z.

Definition 3. Peer-extent Let z be a summary in a given hierarchy of summaries S, and P the set of all peers who participated to the construction of S. The *peer-extent* P_z of the summary z is the subset of peers owning, at least, one record of its extent R_z: $P_z = \{p \in P \mid R_z \cap R_p \neq \emptyset\}$, where R_p is the view over the database of node p, used to build summaries.

Due to the above definition, the notion of *data-oriented* summary in a given database is extended to a *source-oriented* summary in a given P2P network. In other words, summaries can be used as database indexes (e.g. referring to relevant tuples), as well as semantic indexes in a distributed system (e.g. referring to relevant nodes).

The summary hierarchy S will be characterized by its *Coverage* in the P2P system; that is, the fraction of data sources described by S. Relative to the hierarchy S, *Partner Peer* is a peer whose data is described by at least a summary of S.

Definition 4. Partner peers The set of *Partner peers* P_S of a summary hierarchy S is the union of peer-extents of all its summary nodes in S: $P_S = \{\cup_{z \in S} P_z\}$.

For simplicity, in the following we designate by "summary" a hierarchy of summaries maintained in a P2P system, unless otherwise specified.

3 Summary Model for Hierarchical P2P Networks

In this section, we first define the problem of managing data summaries in P2P systems. Second, we describe the architecture of the summary model adopted for hierarchical networks. Then, we present the solutions proposed for summary creation and maintenance.

3.1 Problem Statement

Given a P2P network, we consider the two following assumptions.

- Each peer p owns some tuples (R_p) in a global, horizontally partitioned relation R.
- Users that are willing to cooperate agree on a Background Knowledge BK, which represents their common perception of the domain.

Thus, here the problem of semantic heterogeneity among peers is not addressed, since it is a separate P2P issue on its own. Besides, this work mainly targets collaborative database applications where the participants are supposed to work on "related" data. In such a context, the number of participants is also supposed to be limited, and thus the assumption of a common BK seems not to be a strength constraint. An example of such BK in a medical collaboration is the Systematized Nomenclature of Medicine Clinical Terms (SNOMED CT) [2], which provides a common language that enables a consistent way of capturing, sharing and aggregating health data across specialties and sites of care. On the other hand, the used summaries are data structures that respect the original data schemas [22]. Hence, we can assume that the techniques that have been proposed to deal with information integration in P2P systems (e.g. [15], [20]) can be used here to overcome the heterogeneity of both data and summary representations, in the context of heterogeneous data.

Let $G = (V, E)$ be the graph corresponding to a P2P network of size N, where V is the set of nodes (i.e. $|V| = N$), and E is the set of links between nodes. The ultimate goal is to maintain a global summary that completely describes the global relation R. However, as stated before, the relation R is horizontally partitioned and distributed among autonomous peers. Hence, the

problem can be defined as follows. Given that each peer p_i locally maintains a Local Summary LS_i, we aim to construct the global summary GS_c such that:

$$GS_c = \cup_{i=1}^{N}(LS_i)$$

The local summaries are obtained by integrating the summarization process previously defined into each peer's DBMS. The operator $\cup$ designates the summary merging operation which will be discussed later.

The autonomous and dynamic nature of P2P networks makes the convergence to GS_c quite challenging. It is difficult to build and to keep this summary consistent relative to the current data instances it describes. So, the problem can be redefined as follows.

Given the set of materialized local summaries $\{LS_i, 1 \leq i \leq N\}$, we require to build/materialize the set of global summaries $\{GS_j, 1 \leq j \leq N_G\}$ such that:

- $GS_j = \cup_{i=1}^{l}(S_i)$, where S_i is a local or global summary. Here, the latter is defined as being the merging result of, at least, two summaries (i.e. $l \geq 2$).
- The set of materialized local/global summaries (each having its set of partner peers P_{GS_j}), and the set of links ($E_s \subset E$) between nodes belonging to different sets of partner peers (i.e. links connecting different summaries), provide together an approximation of the *virtual* summary GS_c.

  $$GS_c \approx (\{LS_i, 1 \leq i \leq N_L\}, \{GS_j, 1 \leq j \leq N_G\}, E_s) \quad (1)$$

  N_L is the number of local summaries, which is the number of peers that have not participated to any existing global summary GS_j. While N_G is the number of global summaries built in the network (i.e. $|\cup_{j=1}^{N_G}(P_{GS_j})| + N_L = N$).
- A "good" trade-off should be achieved between the cost of updating the set of materialized summaries and the benefits obtained from exploiting these summaries in query processing.

3.2 Model Architecture

Data indexes are maintained in P2P systems using one of the following approaches. A *centralized* approach maintains a global index over all the data shared in the network, and thus provides a centralized-search facility. A *hybrid decentralized* approach distributes indexes among some specialized nodes (e.g. supernodes), while a *pure decentralized* approach distributes indexes among all the participants in the network (e.g. structured DHTs, Routing Indices). Each of these approaches provides a different trade-off between the cost of maintaining the indexes and the benefits obtained for queries. In [21], the second approach is adopted since it is the only one that exploits peer hetero-

geneity, which is a central key to allow P2P systems scaling up without compromising their decentralized nature [25]. Besides, some also argue that the super-peer networks [7] are the most suited for content-based search [27, 13] since their inherent hierarchical structure allows to build a similar hierarchy over the shared data and metadata.

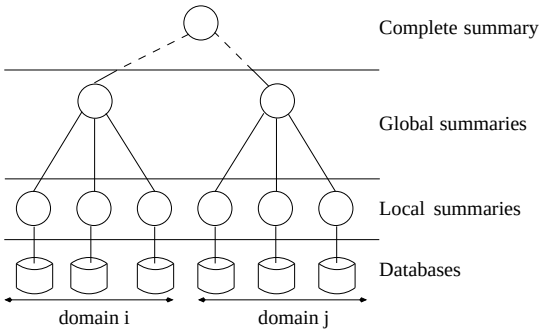

Fig. 4 Model Architecture for Hierarchical P2P Networks

Let us examine the architecture of the summary model which is presented in Figure 4. It is clear that the hierarchy among global summaries follows the hierarchical structure of the underlying network (two summary levels). The network is organized into *domains*, where a domain is defined as being the set of a supernode and its associated leaf nodes. In a given domain, peers cooperate to maintain a global summary over their shared data. The set of global materialized summaries and links between the corresponding domains, provide an approximation of the summary GS_c (Equation 1).

3.3 Summary Management

In this section, we present the algorithms for summary construction and maintenance in a given domain. First, the algorithms are presented in a static context, where all participants remain connected, and then they are extended to cope with peer dynamicity.

3.3.1 Summary Construction

Each global summary GS is associated with a *Cooperation List* (CL) that provides information about its partner peers. An element of CL is composed of two fields. A partner identifier *PeerID*, and a *2*-bit freshness value v that

provides information about the description freshness as well as the availability of the corresponding database.

- value 0 (initial value): descriptions are fresh relative to the original data,
- value 1: descriptions need to be refreshed,
- value 2: original data are not available. This value is used while addressing peer volatility in Section 3.3.3.

Algorithm 1 shows the messages exchanged between peers in order to build a global summary GS. The algorithm starts at a superpeer (referred later as *Summary Peer* SP) who broadcasts a SUMPEER message that contains its identifier, to indicate its ability to host summaries. Since SP is supposed to have high connectivity, a small value of TTL (Time-To-Live) is sufficient to cover a large number of peers (e.g. $TTL = 2$).

The message contains also a hop value h, initialized to 0, which is used to compute the distances between SP and the visited peers. A peer p who received a first SUMPEER message, maintains information about the corresponding summary peer SP (i.e. Line 13). Then, p sends to SP a LOCALSUM message that contains its local summary LS, and thus becomes a partner peer in the SP's domain. Upon receiving this last message, SP merges LS to its current global summary GS, and adds a new element to CL.

However, a peer p who is already a partner may receive a new SUMPEER message. In such a case, only if the new summary peer is nearer than the old one (based on latency), it chooses to drop its old partnership through a DROP message (i.e. Line 11), and it proceeds to participate to a new domain. We now suppose that a peer p does not belong to any domain (i.e. p is not a partner peer), and wants to participate to a global summary construction. Using a selective walk, it can rapidly find a summary peer SP (i.e. FIND message). The information about SP, which is maintained at each of its partners, makes the selective walk even shorter. Once a partner or a summary peer is reached, the FIND message is stopped (i.e. Line 27).

3.3.2 Summary Maintenance

An important issue for any indexing technique is to efficiently maintain the indexes against data changes.

For a local summary, it has been demonstrated that the summarization process guarantees an incremental maintenance (using a *push* mode for exchanging data with the DBMS), while performing with a low complexity. This allows for an online data processing without the need for an overall summary computation.

In this section, a strategy for maintaining *global summary data* is proposed: a global summary GS which has been obtained by merging the local summaries of the set of peers P_{GS}, and its associated cooperation list CL.

Algorithm 1 *Global Summary Construction*

```
1: // Definition of different types of messages
2: SUMPEER=⟨sender⟩ ⟨id, h, TTL⟩; FIND=⟨sender⟩ ⟨h, TTL⟩
3: LOCALSUM=⟨sender⟩ ⟨LS⟩; DROP=⟨sender⟩ ⟨⟩
4: // Treatment of messages
5: Switch msg.type
6: // Receiving information about a summary peer
7: Case (SumPeer):
8:     msg.h++; msg.TTL−−
9:     if (this.SumPeer=null) or (this.SumPeer.h > msg.h) then
10:        if (this.IsPartner) then
11:            Send DROP message to this.SumPeer.id
12:        end if
13:        this.SumPeer:= ⟨msg.id, msg.h⟩
14:        LOCALSUM:= new msg (this.LS); Send LOCALSUM to msg.sender
15:        IsPartner:= True
16:    end if
17:    if msg.TTL > 0 then
18:        Send msg to all neighbors
19:    end if
20: end Case
21: // Searching for a summary peer
22: Case (Find):
23:    msg.TTL−−; msg.h++
24:    if (this.Is_SumPeer) then
25:        PEERSUM:= new msg (this.id, msg.h, 1); Send PEERSUM to msg.sender
26:    else
27:        if (this.SumPeer ≠ null) then
28:            PEERSUM:= new msg (this.SumPeer.id, (msg.h + this.SumPeer.h), 1)
29:            Send PEERSUM to msg.sender
30:        else
31:            if (msg.TTL > 0) then
32:                p′:= highest degree peer in N(p); Send msg to p′
33:            end if
34:        end if
35:    end if
36: end Case
37: // arrival of a new partner
38: Case (LocalSum):
39:    CoopList.add (msg.sender, 0); GlobalSum:= merge (GlobalSum, msg.LS)
40: end Case
41: // departure of a partner
42: Case (Drop):
43:    CoopList.remove (msg.sender)
44: end Case
```

The objective is to keep GS consistent with the current instances of the local summaries.

The latters are supposed to be consistent with the current instances of the original data sources. The maintenance strategy uses both *push* and *pull*

techniques, in order to minimize the number of messages exchanged in the system.

Push: Cooperation List Update

Each peer p in P_{GS} is responsible for refreshing its own element in the cooperation list CL. The partner p monitors the modification rate issued on its local summary LS. When LS is considered as enough modified, the peer p sets its freshness value v to 1, through a push message. The value 1 indicates that the local summary version being merged while constructing GS does not correspond any more to the current instance of the database.

An important feature is that the frequency of push messages depends on modifications issued on local summaries, rather than on the underlying databases. It has been demonstrated in [22] that, after a given process time, a summary becomes very stable. As more tuples are processed, the need to adapt the hierarchy decreases and hopefully, once all existing attribute combinations have been processed, incorporating new tuple consists only in sorting it in a tree. A summary modification may be detected by observing the appearance/disappearance of descriptors in the summary intention.

Pull: Summary Update

The summary peer SP, in its turn, monitors the fraction of old descriptions in GS, i.e. the number of ones in CL. Upon each push message sent to update a freshness value, this fraction value is checked. If it exceeds a threshold value α, the summary update mechanism will be then triggered. Note that the threshold α is our parameter by which the freshness degree of GS will be controlled. In order to update GS, all the partner peers will be pulled to merge their current local summaries into a GS's version. The algorithm is described as follows.

SP initiates a summary update message Sum_Update which is then propagated from a partner to another. This message contains a new summary $NewGS$ (initially empty), and a list of peer identifiers $PeerIDs$ which initially contains the identifiers of all GS's partners (provided by CL). When a partner p receives the Sum_Update message, it first merges $NewGS$ with its local summary and removes its identifier from $PeerIDs$. Then, it sends the message to another partner chosen from $PeerIDs$. If p is the last visited peer (i.e. $PeerIDs$ is empty), it updates the summary data: GS is replaced by the new version $NewGS$, and all the freshness values in CL are reset to zero. This strategy avoids conflicts and guarantees a high availability of the summary data, since only one update operation is performed by the last visited partner.

3.3.3 Peer Dynamicity

In P2P systems, another crucial issue is to maintain the data indexes against network changes. Besides the freshness of summary descriptions, the availability of the original data sources should be also taken into account, given the dynamic behavior of peers.

In unstructured P2P systems, when a new peer p joins the system, it contacts some existing peers to determine the set of its neighbors. If one of these neighbors is a partner peer, p sends its local summary LS to the corresponding summary peer SP, and thus becomes a new partner in the SP's domain. When a partner peer p decides to leave the system, it first sets its freshness value v to two in the cooperation list, through a push message. This value reminds the participation of the disconnected peer p to the corresponding global summary, but also indicates the unavailability of the original data. Here, there are two alternatives to deal with such a freshness value. First, one can keep the data descriptions and use it, when a query is approximately answered using the global summary. A second alternative consists in considering the data descriptions as expired, since the original data are not accessible. Thus, a partner departure will accelerate the summary update initiating. In [21], the authors have adopted the second alternative and consider only a 1-bit freshness value v: a value 0 to indicate the freshness of data descriptions, and a value 1 to indicate either their expiration or their unavailability.

However, if peer p failed, it could not notify its summary peer by its departure. In that case, its data descriptions will remain in the global summary until a new summary update is executed. The update algorithm does not require the participation of a disconnected peer. The global summary GS is reconstructed, and descriptions of unavailable data will be then omitted.

Now, when a summary peer SP decides to leave the system, it sends a release message to all its partners using the cooperation list. Upon receiving such a message, a partner p makes a selective walk to find a new summary peer. However, if SP failed, it could not notify its partners. A partner p who has tried to send push or query messages to SP will detect its departure and thus search for a new one.

4 Query Processing

We describe now how a query Q, posed at a peer p, is processed. The adopted approach consists in querying at first the global summary GS that is available to peer p. Thus, peer p first sends Q to the summary peer SP of its domain, which then proceeds to query the corresponding global summary GS. The type of the query, precise or flexible (when using linguistic terms), and the nature of the returned results allow to distinguish four cases of sum-

mary querying. These cases are illustrated when presenting the two phases of the summary querying mechanism: 1) query reformulation and 2) query evaluation.

4.1 Query Reformulation

First, a precise query Q must be rewritten into a flexible query Q^* in order to be handled by the summary querying process. For instance, consider the following query Q on the Patient relation in Table 1:

```
select age from Patient where sex = ''female''
and BMI < 19 and disease = ''anorexia''
```

This phase replaces the original value of each selection predicate by the corresponding descriptors defined in the Background Knowledge (BK). Therefore, the above query is transformed to Q^*:

```
select age from Patient where sex = ''female'' and
BMI in {underweight,normal} and disease = ''anorexia''
```

The mechanism that assists this query rewriting phase is already defined and used in the mapping service of the summarization process.

Let QS (resp.QS^*) be the *Query Scope* of query Q (resp.Q^*) in the domain, that is, the set of peers that should be visited to answer the query. Obviously, the query reformulation phase may induce false positives in query results. To illustrate, a patient having a BMI value of 20 could be returned as an answer to the query Q^*, while the selection predicate on the attribute `BMI` of the original query Q is not satisfied. However, false negatives cannot occur, which is expressed by the following inclusion: $QS \subseteq QS^*$.

In [21], a user query is supposed to be directly formulated using descriptors defined in the BK (i.e. $Q = Q^*$). As we discussed in the introduction of this work, a doctor that participates to a given medical collaboration, may ask queries like "the *age* of *female* patients diagnosed with *anorexia* and having an *underweight* or *normal* BMI". This assumption eliminates potential false positives that may result from query rewriting. Note that an interface has been defined to allow users to formulate their queries, using linguistic terms, without having knowledge of the pseudo-code language used in the querying mechanism. To illustrate the formulation of queries, recall the following notations:

- R: the schema of the summarized relation
- A: the set of attributes of relation R ($|A| = n$)
- A_i: The i^{th} attribute of relation R
- t: a tuple of the relation R
- z: a summary node of the summary hierarchy S built over the relation R

In the query Q, we distinguish the set X of attributes whose values x are specified by the query predicates, from the set Y of attributes on which the query is projected. In other terms, the general form of query Q is given by: *Select* Y *from* R *where* $X = x$.

For each attribute $A_i \in X$, a "*required characteristic*" is defined as being a linguistic term that appears in the query for attribute A_i. The set C_i of these required characteristics is given by: $C_i = \{d_i^1, d_i^2, \ldots, d_i^m\}$. By extension, the characterization of query Q is the set of k characterizations defined on the attributes of set X: $C = \{C_1, \ldots, C_k\}$, where $k = |X|$. In our example: $X = \{sex, BMI, disease\}$, $Y = \{age\}$, $C_{sex} = \{female\}$, $C_{BMI} = \{underweight, normal\}$, $C_{disease} = \{anorexia\}$, and $C = \{C_{sex}, C_{BMI}, C_{disease}\}$.

Finally, a logical proposition P is associated to a query characterization such that:

$$P = \bigwedge_{i=1}^{k} (\bigvee_{j=1}^{m_i} d_i^j)$$

Intuitively, the presence of multiple required characteristics on a given attribute denotes an alternative, and thus the intra-attribute logical connector is disjunctive (i.e. $C_i = \bigvee_{j=1}^{m_i} d_i^j$). However, the presence of simultaneous characterizations of different attributes implies that the inter-attribute connector is conjunctive (i.e. $P = \bigwedge_{i=1}^{k}(C_i)$). In our example, the query Q is transformed to the proposition P= *(female) AND (underweight OR normal) AND (anorexia)*. In fact, the proposition P represents the canonical form of the query, which will be evaluated by the summary querying process.

4.2 Query Evaluation

First of all, to eliminate any ambiguity, it is worthy to recall that all along our work, we designate by (local/global) summary a hierarchy of summary nodes whose structure has been described in Section 2.4.

This phase deals with matching the set of summary nodes organized in the hierarchy S, against the query Q. A valuation function has been defined to valuate the corresponding proposition P in the context of a summary node z, and thus to determine if z is considered as a result. Then, a selection algorithm performs a fast exploration of the hierarchy and returns the set Z_Q of most abstract summary nodes that satisfy the query. For more details see [26].

Once Z_Q determined, the evaluation process is able to achieve two distinct tasks: 1) Summary answering, and 2) Peer localization.

4.2.1 Approximate Answering

A distinctive feature of the approach presented in this chapter is that a query can be processed entirely in the summary domain. An approximate answer can be provided from summary descriptions, without having to access original database records.

In the result set Z_Q, distinct summary nodes may answer the query in different manners. For instance, the set Z_Q contains summary nodes in which the attribute BMI is described either by *underweight*, or *normal*, or even by the both descriptors. A classification task has been defined in order to regroup the summary nodes in Z_Q according to their characteristics vis-a-vis the query: summary nodes that have the same required characteristics on all predicates (e.g. *sex*, BMI and *disease*) form a class. The aggregation in a given class is a union of descriptors: for each attribute in the selection set Y (i.e. *age*), the querying process supplies a set of descriptors which characterize the summary nodes that answer the query through the same interpretation [26]. For example, according to Table 2, the output set obtained for the different classes found in z_Q is:

- $\{female, underweight, anorexia\} \Rightarrow age = \{adolescent, young\ adult\}$
- $\{female, normal, anorexia\} \Rightarrow age = \{adolescent\}$

In other words, *all* female patients diagnosed with *anorexia* and having an *underweight* or *normal* BMI are *adolescent* and *young* girls. Moreover, the information provided by the tuple count columns may further indicate that almost all of these girls are *adolescent* (i.e. a value of 2.5 for *adolescent*, and a value of 0.5 for *young adult*).

4.2.2 Peer Localization

In centralized environments, returning exact answers to the flexible query Q is simply an extension of the mechanism of returning approximate answers. The extent R_z of a summary node z (see Definition 1) provides the original records that are described by that summary intention. Thus, such a functionality only requires to connect to the underlying database in order to retrieve data.

However, in P2P systems, a summary node may describe tuples that are highly distributed in the network. Therefore, the query Q should be first forwarded to the relevant peers whose databases contain the result tuples. In Definition 3, the *Peer-extent* dimension has been added to a summary node structure, in order to provide the set of peers P_z having data described by its intent.

Here, one can assume that in a given local summary, the extents of its summary nodes are maintained to be able to retrieve raw data from the underlying database. While in a global summary this information is omitted

and only the *Peer-extents* of its summary nodes are maintained to be able to reach the relevant peers in the distributed network.

Based on the *Peer-extents* of summary nodes in the set Z_Q, the set P_Q of relevant peers is defined as follows: $P_Q = \{\cup_{z \in Z_Q} P_z\}$. The query Q is directly propagated to these relevant peers. However, the efficiency of this query routing depends on the completeness and the freshness of summaries, since stale answers may occur in query results. We define a *False Positive* as the case in which a peer p belongs to P_Q and there is actually no data in the p source that satisfies Q (i.e. $p \notin QS$). A *False Negative* is the reverse case in which a p does not belong to P_Q, whereas there exists at least one tuple in the p data source that satisfies Q (i.e. $p \in QS$).

In the case where exact answers are required, suppose now that processing a query Q in a given domain d_i returns C_i results, while the user requires C_t results. We note that, if C_t is less than the total number of results available in the network, Q is said to be a *partial-lookup* query. Otherwise, it is a *total-lookup* query. Obviously, when C_i is less than C_t, the query should be propagated to other domains. To this end, the following variation of the flooding mechanism is adopted.

Let P_i the subset of peers that have answered the query Q in the domain d_i: $|P_i| = (1 - FP) \cdot |P_Q|$, where FP is the fraction of false positives in query results. The query hit in the domain is given by: $(|P_i| \,/\, |d_i|)$. As shown by many studies, the existing P2P networks have small-world features [4]. In such a context, users tend to work in groups. A group of users, although not always located in geographical proximity, tends to use the same set of resources (i.e. *group locality* property). Thus, the probability of finding answers to query Q in the neighborhood of a relevant peer in P_i is supposed to be high (i.e. query answers are supposed to be *nearby*). This probability is also high in the neighborhood of the originator peer p since some of its neighbors may be interested in the same data, and thus have cached answers to similar queries. Such assumptions are even more relevant in the context of interest-based clustered networks. Therefore, the summary peer SP_i of domain d_i sends a flooding request to each peer in P_i as well as to peer p. Upon receiving this request, each of those peers sends the query to its neighbors that do not belong to its domain, with a limited value of TTL. Once a new domain is reached or TTL becomes zero, the query is stopped. Besides, the summary peer SP sends the request to the set of summary peers it knows in the system. This will accelerate covering a large number of domains. In each visited domain, the query is processed as described above. When the number of query results becomes sufficient (i.e. larger than C_t), or the network is entirely covered, the query routing is terminated.

5 Performance Evaluation

In this section, we devise a simple model for the summary management cost. Then, we evaluate that model by simulation using the BRITE topology generator and SimJava.

5.1 Cost Model

A critical issue in summary management is to trade off the summary updating cost against the benefits obtained for queries.

5.1.1 Summary Update Cost

Here, the first undertaking is to optimize the update cost while taking into account *query accuracy*. In the next section, we discuss query accuracy which is measured in terms of the percentage of false positives and false negatives in query results. The cost of updating summaries is divided into: usage of peer resources, i.e. time cost and storage cost, and the traffic overhead generated in the network.

Time Cost

A unique feature of SaintEtiQ is that the changes in the database are reflected through an incremental maintenance of the summary hierarchy. The time complexity of the summarization process is in $O(K)$ where K is the number of cells to be incorporated in that hierarchy [22]. For a global summary update, we are concerned with the complexity of merging summaries. The Merging method that has been proposed is based on the SaintEtiQ engine. This method consists in incorporating the leaves L_z of a given summary hierarchy S_1 into an another S_2, using the same algorithm described by the SaintEtiQ summarization service (referenced in Section 2.3.2). It has been proved that the complexity C_{M12} of the Merging(S_1, S_2) process is constant w.r.t the number of tuples [18]. More precisely, C_{M12} depends on the maximum number of leaves of S_1 to incorporate into S_2. However, the number of leaves in a summary hierarchy is not an issue because it can be adjusted by the user according to the desired precision. A detailed Background Knowledge (BK) will lead to a greater precision in summary description, with the natural consequence of a larger summary. Moreover, the hierarchy is constructed in a top-down approach and it is possible to set the summarization process so that the leaves have any desired precision.

Storage Cost

We denote by k the average size of a summary node z. In the average-case assumption, there are $\sum_{i=0}^{d} B^i = (B^{d+1}-1)/(B-1)$ nodes in a B-arity tree with d, the average depth of the hierarchy. Thus the average space requirement is given by: $C_m = k.(B^{d+1}-1)/(B-1)$. Based on real tests, $k = 512$ bytes gives a rough estimation of the space required for each summary. An important issue is that the size of the hierarchy is quite related to its stabilization (i.e. B and d). As more cells are processed, the need to adapt the hierarchy decreases and incorporating a new cell may consist only in sorting a tree. Hence, the structure of the hierarchy remains stable and no additional space is required. On the other hand, when we merge two hierarchies S_1 and S_2 having sizes of C_{m1} and C_{m2} respectively, the size of the resultant hierarchy is always in the order of the $max\,(C_{m1}, C_{m2})$. However, the size of a summary hierarchy is limited to a maximum value which corresponds to a maximum number of leaves that cover all the possible combinations of the BK descriptors. Thus, storing global summaries does not impose high space storage constraints on the summary peers.

According to the above discussion, the usage of peer resources is optimized by the summarization process itself, and the distribution of summary merging while updating a global summary. Thus, the focus in [21] is on the traffic overhead generated in the P2P network.

Network Traffic

Recall that there are two types of exchanged messages: *push* and *update* messages. Let local summaries have an average lifetime of L seconds in a given global summary. Once L expired, the node sends a (push) message to update its freshness value v in the cooperation list CL. The summary update algorithm is then initiated whenever the following condition is satisfied:

$$\sum_{v \in CL} v/|CL| \geq \alpha$$

where α is a threshold that represents the ratio of old descriptions tolerated in the global summary. While updating, only one message is propagated among all partner peers until the new global summary version is stored at the summary peer SP. Let F_{rec} be the reconciliation frequency. The update cost is:

$$C_{up} = 1/L + F_{rec} \quad \textit{messages per node per second} \tag{2}$$

In this expression, $1/L$ represents the number of push messages which depends either on the modification rate issued on local summaries or the

connection/disconnection rate of peers in the system. Higher is the rate, lower is the lifetime L, and thus a large number of push messages are entailed in the system. F_{rec} represents the number of update messages which depends on the value of α. This threshold is the system parameter that provides a trade-off between the cost of summary updating and query accuracy. If α is large, the update cost is low since a low frequency of update is required, but query results may be less accurate due both to false positives stemming from the descriptions of non existent data, and to false negatives due to the loss of relevant data descriptions whereas they are available in the system. If α is small, the update cost is high but there are few query results that refer to data no longer in the system, and nearly all available results are returned by the query.

5.1.2 Query Cost

When a query Q is posed at a peer p, it is first matched against the global summary available at the summary peer SP of its domain, to determine the set of relevant peers P_Q. Then, Q is directly propagated to those peers. The query cost in a domain d is given by:

$$C_d = (1 + |P_Q| + (1 - FP) \cdot |P_Q|) \;\; messages,$$

where $(1 - FP) \cdot |P_Q|$ represents the query responses messages (i.e. query hit in the domain).

Here we note that, the cooperation list CL associated with a global summary provides information about the relevance of each database description. Thus, it gives more flexibility in tuning the *recall/precision* trade-off of the query answers in domain d. The set of all partner peers P_H in CL can be divided into two subsets: $P_{old} = \{p \in P_H \mid p.v = 1\}$, the set of peers whose descriptions are considered old, and $P_{fresh} = \{p \in P_H \mid p.v = 0\}$ the set of peers whose descriptions are considered fresh according to their current data instances. Thus, if a query Q is propagated only to the set $V = P_Q \cap P_{fresh}$, then precision is maximum since all visited peers are certainly matching peers (no false positives), but recall depends on the fraction of false negatives in query results that could be returned by the set of excluded peers $P_Q \backslash P_{fresh}$. On the contrary, if the query Q is propagated to the extended set $V = P_Q \cup P_{old}$, the recall value is maximum since all matching peers are visited (no false negatives), but precision depends on the fraction of false positives in query results that are returned by the set of peers P_{old}.

Now we consider that the selectivity of query Q is very high, such that each relevant peer has only one result tuple. Thus, when a user requires C_t tuples, we have to visit C_t relevant peers. The cost of inter-domain query flooding is given by:

$$C_f = ((1 - FP) \cdot |P_Q| + 2) \cdot \sum_{i=1}^{TTL} k^i \quad messages,$$

where k is the average degree value (e.g. average degree of 3.5, similar to Gnutella-type graphs). Remember that, the set of relevant peers who have answered the query (i.e. $(1 - FP) \cdot |P_Q|$), the originator and the summary peers participate to query flooding. In this expression, we consider that a summary peer has on average k long-range links to k summary peers. As a consequence, the total cost of a query is:

$$C_Q = C_d \cdot \frac{C_t}{(1-FP)\cdot|P_Q|} + C_f \cdot (1 - \frac{C_t}{(1-FP)\cdot|P_Q|}) \tag{3}$$

In this expression, the term $C_t/((1 - FP) \cdot |P_Q|)$ represents the number of domains that should be visited. For example, when $C_t = ((1 - FP) \cdot |P_Q|)$, one domain is sufficient and no query flooding is required.

5.2 Simulation

The performance of the proposed solutions is evaluated through simulation, based on the above cost model. First, we describe the simulation setup. Then we present simulation results to evaluate various performance dimensions and parameters: scale up, query accuracy, effect of the freshness threshold α.

5.2.1 Simulation Setup

In [21], the SimJava package [10] and the BRITE universal topology generator [1] are used to simulate a power law P2P network, with an average degree of 4. The simulation parameters are shown in Table 3.

Table 3 Simulation Parameters

Parameter	value
Network configuration	
local summary lifetime L	skewed distribution, Mean=3h, Median=1h
number of peers n	16–5000
Workload configuration	
number of queries q	200
matching nodes/query *hits*	10%
System parameter	
freshness threshold α	0.1–0.8

In large-scale P2P file-sharing systems, a user connects mainly to download some data and may then leave the system without any constraint. As reported in [25], these systems are highly dynamic and node lifetimes are measured in hours. As such, data indexes should be mainly maintained against network changes. In collaborative database applications, however, the P2P system is supposed to be more stable. Here, the data indexes should be mainly maintained against data changes, since the shared data may be submitted to a significant modification rate.

As stated before, the work presented in this chapter targets collaborative applications sharing semantically rich data. In simulation tests, synthetic data have been used since it was difficult to obtain/use real, highly distributed P2P databases. Future works aim to employ the summary proposal in the context of astronomical applications, which seem to be attractive because of the huge amount of information stored into the databases. Thus, to provide meaningful results, the performance of the proposed solutions has been evaluated in worst contexts where the data are highly updated. Local summary lifetimes are supposed to follow a skewed distribution with a mean lifetime of 3 hours, and a median lifetime of 60 minutes. Note that summary lifetimes of hours means that the underlying data is submitted to a very high modification rate, since the summaries are supposed to be more stable than original data (as discussed in Section 3.3.2). Besides, such lifetime values allow to predict the performance in large-scale data sharing P2P systems where the rate of node departure/arrival dominates the global summary update initiating.

The tests have been made on moderated-size P2P networks, i.e. the number of peers varies between 16 and 5000. In the used query workload, the query rate is 0.00083 queries per node per second (one query per node per 20 minutes) as suggested in [7]. Each query is matched by 10% of the total number of peers. Finally, our system parameter α varies between 0.1 and 0.8.

5.2.2 Update Cost

This first set of experiments quantifies the trade-off between query accuracy and the cost of updating a global summary in a given domain. Figure 5 depicts the fraction of stale answers in query results for different values of the threshold α. Here, the worst case is illustrated. For each partner peer p having a freshness value equal to 1, if it is selected in the set P_Q then it is considered as false positive. Otherwise, it is considered as false negative. However, this is not the real case. Though it has a freshness value equal to 1, the peer p does not incur stale answers unless its database is changed relative to the posed query Q. Thus, Figure 5 shows the worst, but very reasonable values. For instance, the fraction of stale answers is limited to 11% for a network of 500 peers when the threshold α is set to 0.3 (30% of the peers are tolerated to have old/non existent descriptions).

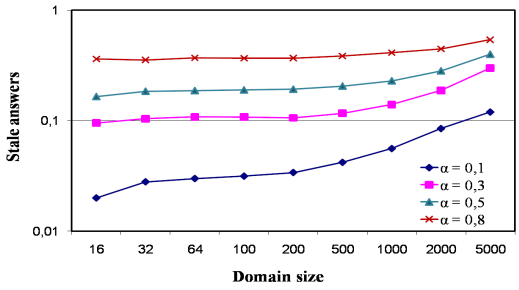

Fig. 5 Stale answers vs. domain size

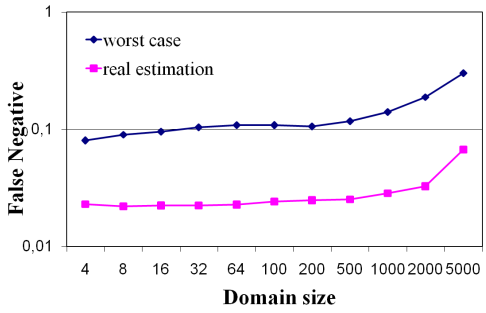

Fig. 6 False negative vs. domain size

As mentioned in Section 5.1.2, if we choose to propagate the query only to the set $V = P_Q \cap P_{fresh}$ we eliminate the possible false positives in query results. However, this may lead to additional false negatives. Figure 6 shows the fraction of false negatives in function of the domain size. Here, for a peer having a freshness value equal to 1, the probability of the database modification relative to the issued query is taken into account. We see that the fraction of false negatives is limited to 3% for a domain size less than 2000 (i.e. network size less than 8000). The real estimation of stale answers shows a reduction by a factor of 4.5 with respect to the preceded values.

Figure 7 depicts the update cost in function of the domain size, and this for two threshold values. The total number of messages increases with the domain size, but not surprisingly, the number of messages per node remains almost the same. In the update cost equation 2, the number of push messages for a given peer is independent of domain size. More interestingly, when the threshold value decreases (from 0.8 to 0.3) we notice a little cost increasing of 1.2 on average. For a domain of 1000 peers, the update cost increases from 0.01056 to 0.01296 messages per node per minute (not shown in figure). However, a small value of the threshold α allows to reduce significantly the fraction of stale answers in query results, as seen in Figure 5. We conclude

therefore that tuning the system parameter, i.e. the threshold α, do not incur additional traffic overhead, while improving query accuracy.

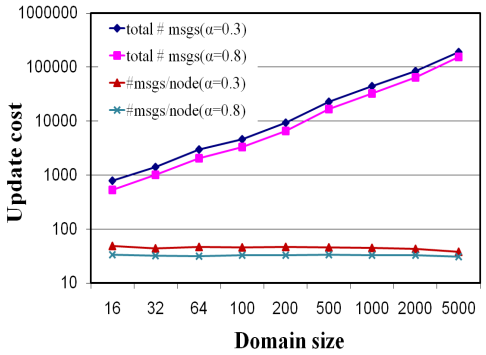

Fig. 7 number of messages vs. domain size

5.2.3 Query Cost

This set of experiments compares the algorithm for query processing against centralized-index and pure non-index/flooding algorithms. A centralized-index approach is very efficient since a single message allows locating relevant data. However, a central index is vulnerable to attack and it is difficult to keep it up-to-date. Flooding algorithms are very used in real life, due to their simplicity and the lack of complex state information at each peer. A pure flooding algorithm consists in broadcasting the query in the network till a stop condition is satisfied, which may lead to a very high query execution cost. Here, the flooding is limited by a value 3 of TTL.

According to Table 3, the query hit is 10% of the total number of peers. For the query processing approach, which is mainly based on summary querying (SQ), it is considered that each visited domain provides 10% of the number of relevant peers (i.e. 1% of the network size). In other words, we should visit 10 domains for each query Q. From equation 3, we obtain: $C_Q = (10 \cdot C_d + 9 \cdot C_f)$ messages. Figure 8 depicts the number of exchanged messages to process a query Q, in function of the total number of peers. The centralized-index algorithm shows the best results that can be expected from any query processing algorithm, when the index is complete and consistent, i.e. the index covers the totality of data available in the system, and there are no stale answers in query results. In that case, the query cost is: $C_Q = 1 + 2 \cdot ((0.1) \cdot n)$ messages, which includes the query message sent to the index, the query messages sent to the relevant peers and the query response messages returned to the originator peer p.

In Figure 8, we observe that the algorithm *SQ* shows good results by significantly reducing the number of exchanged messages, in comparison with a pure query flooding algorithm. For instance, the query cost is reduced by a factor of 3.5 for a network of 2000 peers, and this reduction becomes more important with a larger-sized network. We note that in these tests, the worst case of the *SQ* algorithm is considered, in which the fraction of stale answers of Figure 5 occurs in query results (for $\alpha = 0.3$).

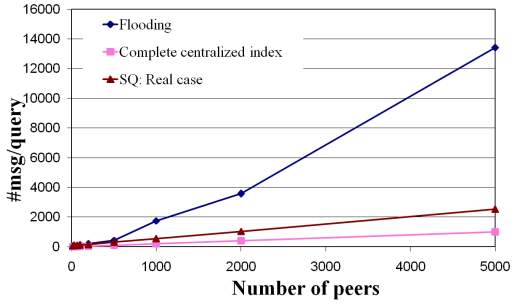

Fig. 8 Query cost vs. number of peers

6 Related work

Current works on P2P systems aim to employ *content-based* routing strategies, since the content of data can be exploited to more precisely guide query propagation. These strategies require gathering information about the content of peer's data. However, the limits on network bandwidth and peer storage, as well as the increasing amount of shared data, call for effective summarization techniques. These techniques allow exchanging compact information on peer's content, rather than exchanging original data in the network.

Existing P2P systems have used keyword-based approaches to summarize text documents. For instance, in [3] documents are summarized by keyword vectors, and each node knows an approximate number of documents matching a given keyword that can be retrieved through each outgoing link (*i.e.* Routing Indices *RI*s). Although the search is very bandwidth-efficient, *RI*s require flooding in order to be created and updated, so the method is not suitable for highly dynamic networks. Other works (e.g. [9]) investigate Vector Space Model (VSM) and build *Inverted Indexes* for every keyword to cluster content. In this model, documents and queries are both represented by a vector space corresponding to a list of orthogonal term vectors called *Term Vector*. The drawback of VSM is its high cost of vector representations in case of P2P churns. In [13], a *semantic-based* content search consists in

combining VSM to Latent Semantic Index (LSI) model to find semantically relevant documents in a P2P network. This work is based on hierarchical summary structure over hybrid P2P architecture, which is closely related to what we are presenting in this paper. However, instead of representing documents by vector models, we describe structured data (*i.e.* relational database) by synthetic summaries that respect the original data schema.

To the best of our knowledge, none of the P2P summarization techniques allows for an *approximate query answering*. All works have focused on facilitating content-based query routing, in order to improve search efficiency. We believe that the novelty of the approach proposed in this chapter relies on the double exploitation of the employed data summaries. These summaries allow for a semantic-based query routing, but also to approximately answer the query using their intentional descriptions.

7 Conclusion

This chapter proposed a model for summary management in hierarchical P2P systems. The innovation of this proposal consists in combining the P2P and database summarization paradigms, in order to support data sharing on a world wide scale. The database summarization approach that we proposed provides efficient techniques for data localization as well as for data description in P2P systems. In fact, the produced summaries are semantic indexes that support locating relevant data based on their content. Besides, an important feature is that these summaries are compact data descriptions that can approximately answer a query without retrieving original records from huge, highly distributed databases.

This work made the following contributions. First, an appropriate summary model for hybrid P2P systems is proposed. Then, efficient algorithms for summary management are described, and a query processing mechanism that relies on summary querying is presented. Performance evaluation showed that the cost of query routing in the context of summaries is significantly reduced in comparison with flooding algorithms, without incurring high costs of summary maintenance.

In the summary model for hierarchical P2P networks [21], it has been assumed that peers are grouped around high-connectivity nodes. However, to better exploit the *data locality* property of most of current P2P systems, future works intend to study the organization of nodes based on similarity between their summaries. Ongoing works on the SaintEtiQ model aim to define a similarity distance between summaries. However, such a proposal requires a complete study of the obtained clustering scheme, i.e. the number of clusters, the cluster sizes, the stability against node dynamics.

Besides, one can notice that the proposed summaries can serve data anonymization purposes. To illustrate, a given hospital which is participat-

ing to a medical collaborative application may first perform an obfuscation of its database through the generation of a local summary. Then, the different participants exchange and share such intelligible summaries, represented in a higher abstraction level. This allow to learn the "essential" from a given database without revealing personal information about patients. A future work plans to study how these summaries can be also used to make a data source anonymization. In fact, due to the peer-extent information, a peer may reveal the source providing a given summary while executing the summary update algorithm. In some cases, participants may also prefer hiding the characterization of their data. Hence, a given participant can exploit data summaries of other participants without being able to precise which source is providing a specified summary.

References

1. http://www.cs.bu.edu/brite/
2. http://www.snomed.org/snomedct
3. A.Crespo, H.G.Molina: Routing indices for peer-to-peer systems. In: Proc. of the 28 tn Conference on Distributed Computing Systems (2002)
4. A.Iamnitchi, M.Ripeanu, I.Foster: Locating data in (small-world?) peer-to-peer scientific collaborations. In: IPTPS, pp. 232–241 (2002)
5. A.Shoshani: OLAP and statistical databases: Similarities and differences. In: Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 185–196. ACM Press (1997)
6. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Commun. ACM **18**(9), 509–517 (1975). DOI http://doi.acm.org/10.1145/361002.361007
7. B.Yang, H.G.Molina: Comparing hybrid peer-to-peer systems. In: Proc VLDB (2001)
8. Comer, D.: Ubiquitous b-tree. ACM Comput. Surv. **11**(2), 121–137 (1979). DOI http://doi.acm.org/10.1145/356770.356776
9. F.Cuenca-Acuna, C.Peery, R.Martin, T.Nguyen: Planetp: Using gossiping to build content addressable peer-to-peer information sharing communities. In: HPDC-12 (2003)
10. F.Howell, R.McNab: Simjava: a discrete event simulation package for java with the applications in computer systems modeling. In: Int. Conf on Web-based Modelling and Simulation, San Diego CA, Society for Computer Simulation (1998)
11. Han, J., Fu, Y., Huang, Y., Cai, Y., Cercone, N.: Dblearn: A system prototype for knowledge discovery in relational databases. In: Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, May 24-27, 1994, p. 516. ACM Press (1994)
12. H.Jagadish, R.Ng, B.Ooi, A.Tung: Itcompress: An iterative semantic compression algorithm. In: 20th International Conference on Data Engineering, p. 646 (2004)
13. H.Shen, Y.Shu, B.Yu: Efficient semantic-based content search in p2p network. IEEE Transactions on Knowledge and Data Engineering **16**(7) (2004)
14. Ioannidis, Y.: The history of histograms (abridged). In: VLDB '2003: Proceedings of the 29th international conference on Very large data bases, pp. 19–30. VLDB Endowment (2003)
15. I.Tartinov, *et al*: The Piazza peer data management project. In: SIGMOD (2003)

16. K.Thompson, P.Langley: Concept formation in structured domains. In: Concept formation: Knowledge and experience in unsupervised learning, pp. 127–161. Morgan Kaufmann
17. L.A.Zadeh: Concept of a linguistic variable and its application to approximate reasoning-I. Information Systems **8**, 199–249 (1975)
18. M.Bechchi, G.Raschia, N.Mouaddib: Merging distributed database summaries. In: ACM Sixteenth Conference on Information and Knowledge Management (CIKM) (2007)
19. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical recipes in C (2nd ed.): the art of scientific computing. Cambridge University Press, New York, NY, USA (1992)
20. R.Akbarinia, V.Martins, E.Pacitti, P.Valduriez: Design and implementation of appa. In: Global Data Management (Eds. R. Baldoni, G. Cortese and F. Davide). IOS press (2006)
21. R.Hayek, G.Raschia, P.Valduriez, N.Mouaddib: Summary management in p2p systems. In: EDBT, pp. 16–25 (2008)
22. R.Saint-Paul, G.Raschia, N.Mouaddib: General purpose database summarization. In: Proc VLDB, pp. 733–744 (2005)
23. Ruspini, E.H.: A new approach to clustering. Information and Control **15**, 22–32 (1969)
24. S.Babu, G.Minos, R.Rajeev: Spartan: A model-based semantic compression system for massive data tables. In: Proc. of the 2001 ACM Intl. Conf. on Management of Data (SIGMOD), pp. 283–295 (2001)
25. S.Saroiu, P.Gummadi, S.Gribble: A measurement study of peer-to-peer file sharing systems. In: Proc of Multimedia Computing and Networking (MMCN) (2002)
26. W.A.Voglozin, G.Raschia, L.Ughetto, N.Mouaddib: Querying the SaintEtiQ summaries–a first attempt. In: Int.Conf.On Flexible Query Answering Systems (FQAS) (2004)
27. W.Nejdl, W.Siberski: Design issues and challenges for rdf- and schema-based peer-to-peer systems. SIGMOD Record **32**, 2003 (2003)